REMARKS/ARGUMENTS

In the Advisory Action dated November 19, 2004, the Examiner denied entry of Applicant's response of October 12, 2004, to the Final Office Action of July 14, 2004. Applicants thus request entry of the amendment and remarks presented in this Preliminary Amendment. Claims 1-14 remain pending in this application and stand rejected. Claim 15 is canceled rendering its rejection moot. Claims 1-14 were rejected in the Final Office Action of July 14, 2004 under 35 U.S.C. 112, first paragraph, as containing subject matter which was not described in the application in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claims 1-2, 4-6, 8-12, and 14 were rejected in the Final Office Action of July 14, 2004, under 35 U.S.C. 103(a) as being unpatentable over Lauterbach, "Accelerating Architectural Simulation by Parallel Execution of Trace Samples", Tech. Report SMLI TR-93-22, Sun Microsystems Laboratories, Inc., December 1993, pages 1-14, (hereinafter Lauterbach) in view of Applicant's assertions as shown in Fig. 1A.

Claims 1-3 and 11-13 are amended. Support for these amendments is provided, for example, in Figs. 3A, 3B, page 8, lines 5-19 and 32-33, page 9, lines 2-6 and 25-31, as reproduced below:

> "Figure 3A is a block diagram of one type of system 300, which includes program 210, checkpoints 310, 320, 330, code fragments 314, 324, 334, runs 318, 328, 338 on low level simulators 252, 254, 256, functional data 312, 322, 332, and performance data 316, 326, 336. In a specific embodiment, checkpoints 310, 320, 330 cover program 210 entirely. In a further embodiment, checkpoints 310, 320, 330 divide program 210 into code fragments 314, 324, 334 of determined lengths. In a specific embodiment, the determined length code fragments are equal length code fragments.
>
> In another embodiment, program 210 is executed on each of low level simulator 252, 254, 256 up to a certain point, where each certain point is the next checkpoint immediately following the corresponding checkpoint, 310, 320, 330. For example, program 210 is run on low level simulator 252, beginning at checkpoint 310 and ending at the next checkpoint, which is checkpoint 320. Runs 318, 328, 338 signify the executions of low level simulators 252, 254, 256.

In another embodiment, the certain point is a point in program 210 before or after the next checkpoint immediately following the corresponding checkpoint, 310, 320, 330 but after the corresponding checkpoint." (page 8, lines 5-19)

"Figure 3B is a block diagram of another type of system 350, which includes program 210, checkpoints 360, 370, 380, code fragments 364, 374, 384, runs 368, 378, 388 on low level simulators 252, 254, 256, functional data 362, 372, 382, and performance data 366, 376, 386. In a specific embodiment, checkpoints 360, 370, 380 cover random parts of program 210. In a further embodiment, checkpoints 360, 370, 380 divide program 210 into code fragments 364, 374, 384 of random lengths. In another embodiment, code fragments 364, 374, 384 do not overlap. In a different embodiment, code fragments 364, 374, 384 are not connected.....

This could be especially true in a specific embodiment where checkpoints 360, 370, 380 cover random parts of program 210, where checkpoints 360, 370, 380 divide program 210 into code fragments 364, 374, 384 of random lengths, where code fragments 364, 374, 384 do not overlap, where code fragments 364, 374, 384 are not connected, and where program 210 is executed on each of low level simulators 252, 254, 256 up to a certain point, where each certain point is a point in the program a random length after the corresponding checkpoint 360, 370, 380" (page 8, lines 32-33, page 9, lines 2-6 and 25-31)

In view of the foregoing amendments and following remarks, reconsideration of rejections of claims 1-14 is respectfully requested.

Rejections Under 35 U.S.C. 112

In rejecting claims 1-15 under 35 U.S.C. 112, first paragraph, the Examiner asserts:

"For example, as shown in Fig. 3A and Fig. 3B and described in the corresponding specification, each low level simulator run one code fragment which may be of determined length or random length. However, without undue experiment, it is unclear what will happen if the destination of a branch instruction is outside of its current code fragment."

Claim 1 is amended to recite, in part, "....dividing the program into a plurality of independent code fragments wherein a destination branch of an instruction in each code fragment falls within that code fragment; thereafter establishing a plurality of checkpoints; wherein each of the plurality of checkpoints is established along one of a beginning point or an ending point of

a different one of the code fragments; thereafter....." Consequently, in accordance with claim 1, before the check points are established, the program is divided into a plurality of code fragments such that the destination branch of an instruction in each code fragment falls within that code fragments. Therefore the condition specified by the Examiner and set for the below:

> "how to establish checkpoints such that the destination of any branch instruction will not be outside of its code fragment"

does not occur. Accordingly, withdrawal of the rejections of claims 1-14 under 35 U.S.C. 112, first paragraph, is respectfully requested.

Rejections Under 35 U.S.C. 103

Claims 1-2, 4-6, 8-12, and 14-15 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Lauterbach, in view of Applicant's assertions as shown in Fig. 1A. Regarding claim 1, the Examiner asserts:

> " Lauterbach discloses a method for validating performance and functionality of a processor, comprising the steps of:
> (Claim 1) executing a program on a high level simulator of said processor (the entire execution of the sampled program, page 3, paragraph 3);
> establishing a plurality of checkpoints (the start of the sample instruction trace, page 3, paragraph 3, fifty samples, page 3, paragraph 2);
> saving state data at each of said check points (initial cache state, page 3, paragraph 3)
> running said program on a plurality of simulators of said processor in parallel, starting each of said simulators at a corresponding checkpoint with corresponding state data associated with said corresponding checkpoint (parallel execution, page 10, section 5.0)...."

Applicants respectfully traverse this rejection for at least the following reasons.

Lauterbach is concerned with evaluating processor architectural performances and not with validating the functionality of a processor. For example, Lautrebach discloses:

> "We have developed a viable technique for accelerating architectural simulation to enable number of architectural tradeoffs to be investigated in a short period of time" (page 11, paragraph 4)
>
> "In order to quickly decide which architectural features are to be included in future processors, we have developed a simulation approach that uses the samples of benchmark program instruction traces..." (abstract)

As is known to those skilled in the art, validating the functionality of a processor is a process that is carried out after the architectural evaluation and design of the processor is, in large part, finalized. Applicants thus submit that Lauterbach, whether taken singly, or in combination with Applicant's assertions, as shown in Fig 1 and described in pages 1-12 of the original disclosure, fails to disclose "generating functional data to validate functionality of the processor", as recited, in part, in claim 1. Claim 1 is thus allowable over Lauterbach in view of Applicant's assertions.

Claim 1 is further allowable over Lauterbach in view of Applicant's assertions for reciting, in part, "executing instructions in said program..... with corresponding state data associated with said corresponding checkpoint." As best understood, no instructions are executed in Lauterbach's simulations since there is no data associated with the instructions. To reduce simulation times, Lauterbach discloses using "samples of the benchmark program so that only a small portion of the entire instruction trace needs to be simulated". (page 2, paragraph 2). The "start of the sample instruction trace" in Lauterbach is an acknowledgement of the initiation of the tracing of the instructions in the program sample. For example, on page 5 Lauterbach provides:

> "The SPARC instruction tracer (SHADE(6)) produces a 14-byte structure to describe each instruction traced. For each instruction, the following information is provided:
>
> - program counter (4 bytes)
> - Instruction (4 bytes)
> - Effective memory address (4 bytes)
> - Annulment flag (4 bytes)

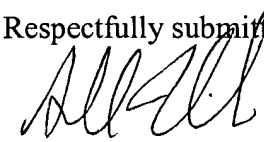
- Branch direction flag (1 byte)"

As is seen from this excerpt, there is no data associated with tracing of instructions. Applicants thus submit that Lauterbach, whether taken singly, or in combination with Applicant's assertions, as shown in Fig 1 and described in pages 1-12 of the original disclosure, fails to disclose "executing instructions in said program..... with corresponding state data associated with said corresponding checkpoint", as recited, in part, in claim 1. Claim 1 is thus further allowable over Lauterbach whether taken singly, or in combination with Applicant's assertions, as shown in Fig 1 and described in pages 1-12 of the original disclosure.

Claims 2-10 are dependent on claim 1 and are thus allowable for at least the same reasons as is claim 1. Claims 11-14 are allowable for at least the same reasons as is claim 1.

In view of the foregoing, Applicants believe all claims now pending in this Application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested. If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 650-326-2400.

Respectfully submitted,



Ardeshir Tabibi
Reg. No. 48,750

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: (650) 326-2400
Fax: (650) 326-2422
AT:deh

60394940 v1